Départ. Informatique
LP DEVOPS - Module Conception d'Architectures Logicielles
Fév 2023

V. Deslandres

TD 4 - Design Patterns (2)

EXERCICE 1 – Quel pattern ?

Dans cette application, on effectue des recherches sur des mots-clefs. On a donc une interface Search :

```
public interface Search {
    String searchFor(String... keywords) throws Exception;
}
```

On définit une première classe d'implémentation de **`Search`** :

```
public class OneSearch implements Search {
    @Override
    public String searchFor(String... keywords) {
        return "Result: " + String.join(" ", Arrays.asList(keywords));
    }
}
```

Puis une deuxième classe d'implémentation de **`Search`** :

```
public class ControlledSearch implements Search {
    private static Set<String> bannedSearches = new
HashSet<>(Arrays.asList("China", "Russia", "Iran", "Pakistan"));
    private OneSearch searchService;

public ControlledSearch() {
        this.searchService = new OneSearch();
    }
    @Override
    public String searchFor(String... keywords) throws Exception {
        for (String keyword : keywords) {
           if (bannedSearches.contains(keyword.trim().toLowerCase())) {
               throw new Exception(keyword.toUpperCase() + " is a
restricted keyword!");
            }
        }
        return this.searchService.searchFor(keywords);
    }
}
```

Notre application n'utilise que **`ControlledSearch`**. Quel pattern avons-nous utilisé dans cette deuxième classe ?

EXERCICE 2 – Festival de musiques

Vous êtes chargé de proposer le S.I. d'une association qui organise des festivals (musique, théâtre, etc.). Si chaque festival possède ses particularités (lieu, dates, façon de communiquer, etc.), la préparation des festivals regroupe des tâches sont communes : gérer les participants, faire la programmation, définir le budget, rechercher des financements, communiquer, réserver le lieu, monter les scènes, réserver le SAMU, gérer les bénévoles, faire le bilan financier, etc.

On voudrait donc pouvoir créer des festivals *locaux (concrets)* à partir d'un festival *générique (abstrait)*, pour lequel on définit toutes les tâches communes à réaliser par ce festival (c'est le 'contrat', certaines méthodes étant abstraites, par ex. : réserver le lieu). Ce contrat sera modifié (ou implémenté) au niveau des festivals concrets.

On imagine ainsi créer une classe *AssociationFestival*, avec une méthode générale *créerFestival (String type)* qui va renvoyer un *Festival*, qui est valable pour tous les festivals. Les types de festival gérés à ce jour sont : **inDoor** et **outDoor**. Chaque type de festival **inDoor** va être capable de créer un 'vrai' festival selon le type donné en paramètre : Jazz à Vienne sera par ex. défini comme un festival **outDoor** de musique *Jazz*.
Pour un festival de musique on enregistre le genre, et on réserve un lieu. Pour un festival **inDoor** de musique classique, on devra en plus vérifier l'acoustique de la salle. Pour un festival **inDoor** pour les jeunes, on doit obtenir une autorisation spéciale et définir la plage des âges concernés.

Le festival est caractérisé par son nom, une ville et un lieu, une taille (on distingue les *gros* festivals des *moyens*, et aussi les *petits* : utile pour les filtres), une date de début et de fin, un budget souhaité, un budget alloué, un montant des dépenses, et qui regroupe des participants (les groupes), des mécènes et des sponsors, et est en relation avec d'autres associations partenaires.

2.1- Créer la classe abstraite *Festival* avec tous les attributs que vous pensez utile et les relations avec les classes nécessaires. Ajoutez le contrat tel que mentionné dans l'énoncé (que vous pouvez compléter : réserverLieu(), gérerParticipants(), définirBilletterie(), etc.).

2.2- Créer les sous-classes des festivals 'concrets' : OutDoorMusique, InDoorMusique, InDoor Classique, InDoorJeunes et InDoorThéatre, avec leurs spécificités mentionnées dans l'énoncé.

2.3- Créer maintenant une classe abstraite AssociationFestival, définie par son président, co-président, trésorier et secrétaire, et qui propose 2 méthodes :

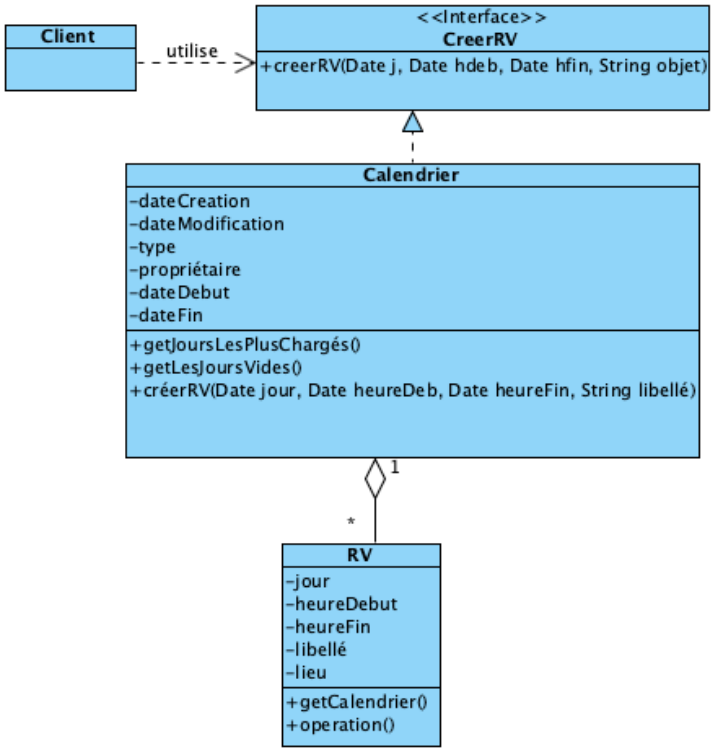
- La méthode *préparerFestival(String type)* qui consiste, à partir d'une instance de Festival créée localement, à appliquer toutes les 'méthodes du contrat' ;
- La méthode abstraite *créerFestival(String type)* retourne une instance de la classe Festival.
- Bien entendu *préparerFestival()* fait appel à *créerFestival().*

2.4- Quel design pattern avons-nous utilisé ici ?

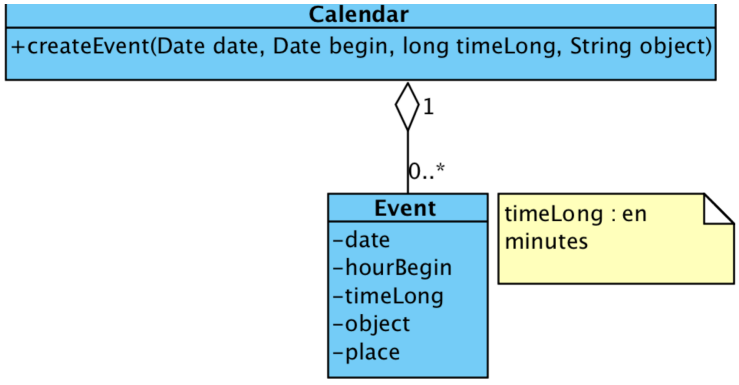
EXERCICE 3 – Calendrier avec des rendez-vous

Au sein de votre équipe, pour la majorité des applications, on utilise un Calendrier composé de rendez-vous.

La méthode *créerRV()* est une implémentation d'une interface générique CreerRV, c'est cette dernière qu'utilisent toutes les applications :



Voilà qu'un nouveau (gros) client vous demande d'interfacer votre logiciel phare avec son propre calendrier, et qui possède une structure similaire mais légèrement différente :



Date begin est l'heure de début de l'événement.

Que faut-il faire pour intégrer ce calendrier à votre application de manière à satisfaire le Client ? Donnez le DCL UML.